DEVELOPMENT OF PC-BASED SPACECRAFT SIMULATOR FOR EOS GROUND SYSTEM TESTING

Estelle S. Noone, Kevin Parker, Janice Swope
Computer Sciences Corporation, Goddard Space Flight Center, Maryland

ABSTRACT

Spacecraft communication simulators are extremely useful for integration and testing of spacecraft control centers and supporting ground systems. To reduce development costs, a Windows NT PC-based simulation system is being developed to support testing for upcoming NASA missions. The spacecraft simulation suite of tools integrates modules within a core infrastructure and is customized to meet mission unique specifications not met by the baseline system.

KEY WORDS

Spacecraft simulation, command and telemetry simulator

INTRODUCTION

Spacecraft communication simulators and test tools have long played an important role in pre-launch test support for NASA missions. Spacecraft simulators are routinely used during development of control centers, integration and test of ground system elements, ground system readiness verification, end-to-end testing, and flight operations team test and training activities.

Today, NASA budgets are shrinking and competition among projects for limited funding dictates the need to develop ways to do more, at less cost, in a shorter time. The NASA community has been challenged by NASA Administrator Dan Goldin to build systems, develop approaches, and work "Better, Faster, Cheaper" than in the past.

This challenge applies to developing spacecraft simulators as well. Success in this environment will require a new generation of lower cost, higher quality spacecraft simulators and test tools. Advances in both hardware systems and software methodologies provide the opportunity to make this happen.

Typically, spacecraft simulators have been developed on a mission-by-mission basis because of the many differences among spacecraft manufacturers' designs. The ultimate goal of a purely generic or universal spacecraft simulator is reasonable but will not be easily attained until spacecraft architectures and data communication protocols become more standardized. Until then, a short-term goal is to develop a framework and a core system that requires minimal rework for subsequent missions with similar requirements.

Mass-market, personal computers (PCs) can now provide computing speed and near-real-time responsiveness sufficient for spacecraft and ground system simulations. A PC platform with Windows NT or Linux operating systems provides a basis for development and operations at a fraction of the cost of previous systems. Even equipped with specialized communications and timing boards, a PC costs only about $5K to $10K; the previous VME systems used as the standard simulator platform in the Goddard Space Flight Center (GSFC) Simulations Center of Excellence (SCE) have a base system cost of $50K to $60K.

In addition, software methodology advances, particularly in the object-oriented domain, facilitate development of modular, scalable systems with even greater levels of reuse than have been demonstrated in the past. Moreover, integrated development environments and design and development tools aid in code generation and general software support to increase productivity gains further.

This paper describes a new generation of spacecraft communications simulators being developed at the GSFC in Greenbelt, Maryland, to reduce long-term development costs and meet NASA’s needs now and into the next millennium. The paper focuses specifically on the simulators being developed for the Earth Observing System (EOS) project and briefly addresses the approach taken to develop the new simulator system.

BACKGROUND

The SCE has evolved from more than 20 years of experience developing spacecraft simulators and other real-time communications test tools to test NASA ground systems for numerous missions at GSFC and other NASA centers. The SCE has a long history of producing small, relatively inexpensive systems quickly, from the system used in the 1983 NASA Tracking Data and Relay Satellite (TDRS) rescue to the QuikSCAT simulator recently developed in less than 1 month. The SCE has developed complex systems as well, such as the high-fidelity Landsat-7 spacecraft simulator and a whole range of test tools that support the Hubble Space Telescope and its control center.

Simulators mean different things to different users. The SCE spacecraft simulators accept and respond to commands, generating telemetry realistic enough for test purposes to convince the ground systems and the flight operations team (FOT) that they are communicating with a satellite in space. SCE systems simulate satellite telemetry from space, before the real spacecraft is launched, to allow control center development and testing, ground system testing, and FOT training and procedure validation during spacecraft construction. Spacecraft simulators also allow users to create anomalous conditions, which are useful for training and would be too dangerous to test using the real spacecraft. The simulators usually reside in the control centers or in simulations and test facilities. Portable systems are taken to NASA tracking stations and launch sites to support remote communications testing.

Simulator fidelity can be low, medium, or high:

- A low-fidelity simulator provides basic command receipt and telemetry transmission capabilities, with mostly static telemetry data and limited command responses. A project database (PDB) is not used.

- A medium-fidelity simulator uses the PDB as its basis for command responses and telemetry generation. A low-end system does little more, whereas a high-end system adds models of internal spacecraft subsystems to provide additional command responses and telemetry values.

- A high-fidelity simulator has command handling, telemetry generation, and modeling capabilities that are largely based on spacecraft functionality rather than generic simulation functionality. Telemetry is heavily populated, and almost all command responses, both direct and indirect, are generated.

The low- to medium-fidelity simulators are the most common, which usually reflects the limited funding typically available for test tools. These simulators are generally used for control center and ground system test support. The medium- to high-fidelity range simulators provide all the basic capabilities as well as a good approximation of the spacecraft; they are used for FOT training.

Code reuse and building upon a core, baseline system is not a new concept. Within a similar architecture and class of simulators, SCE reuse has historically averaged 70 to 80 percent. During the SCE's existence, spacecraft simulator architectures have evolved through several distinct generations. The SCE used to write operating systems and programs in assembly language, which ran on 32K Rolm 1602 and Data General computers. The SCE's most recent generation of simulators ran on VME platforms under the PDOS and Unix operating systems and were written in C. One of the last simulators developed within this class was for the Earth Observing System (EOS) AM-1 spacecraft, currently scheduled for launch in August 1999.

The time has come for a platform and architecture migration. The next generation of simulators, which the SCE calls the Scalable, Integrated, Multimission Simulation Suite (SIMSS), builds on SCE expertise and exploits advances in hardware and software technologies to reduce development costs. The first simulator being built using this next generation SIMSS architecture is for the EOS PM-1 satellite; it is referred to as SIMSS/PM.

EOS SPACECRAFT SIMULATORS

The EOS project is an ambitious one. Its mission life is expected to stretch 15 to 20 years, with a series of spacecraft all intended to measure various indicators of global climate change. The first spacecraft, EOS AM-1, was to have launched in June 1998; however, problems with the control center, ground system, spacecraft, and most recently the launch vehicle have caused several slips.

The first release of the EOS AM-1 simulator, called the Multimode Portable Simulator (MPS), was delivered in December 1996; six more releases, with incremental capabilities, have been delivered since then. The sixth release was in January 1999. The MPS is considered a medium-fidelity simulator. It simulates the low rate, non-science telemetry data formats, and receives and validates commands from the control center. It uses the AM-1 project-supplied database for telemetry generation and command validation. It was primarily used as a test tool for the development of the EOS control center (EOC). It was also used as a data source for testing the interface with the EOS Data and Operation System (EDOS), which serves as the EOS data capture and processing facility, as well as for testing and trouble shooting with the EOS ground system and communications network.

The MPS is actually two simulators in one, hence the multimode reference in its name. The MPS supports both serial clock and data and Internet Protocol (IP) interfaces for command and telemetry data. As shown in Figure 1, it simulates the spacecraft serial data formats to and from EDOS. It also simulates some EDOS functions and supports the IP data formats across the EDOS - EOC interface, which involves data reformatting and additional headers.

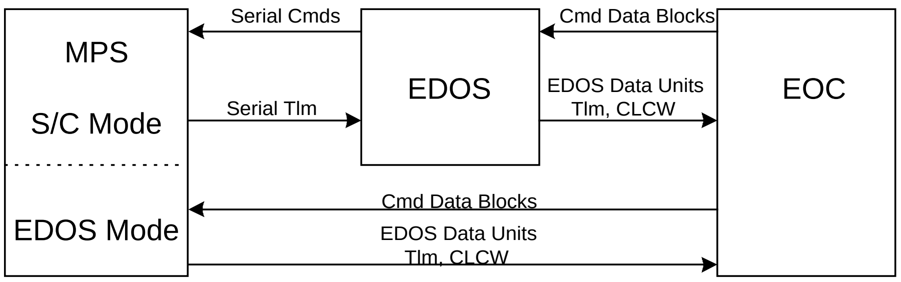

Figure 1. MPS: Two Simulators in One

The MPS uses a client-server architecture. The user interface resides on a Unix-based processor and was created new for this application. The simulator engine runs on a MVME 68030 processor under the PDOS operating system and was built on heritage systems developed for previous simulators used on the XTE and TRMM spacecraft projects. The cost of the hardware, operating system, and development environment was $90K per unit, exclusive of developing the actual software. The application, which is about 60,000 lines of code, is written in C.

Besides having expensive hardware, the basic system was neither easily scalable nor extensible. The start-up procedure for the two processors and associated input/output (I/O) cards was cumbersome and prone to synchronization problems when the system was used remotely and not shut down properly. The target VME platform served a single purpose—hosting the simulator. Furthermore, the software development environment was very limited with primitive tools. Developers pulled code from existing simulators and made changes for the AM-1 spacecraft. Years of inheriting and then modifying code from predecessors had produced inefficiencies in the software. No commercial, off-the-shelf (COTS) products were used in the application, and the real-time operating system was specialized and lacked common features of the more mainstream systems.

Despite the less than hospitable hardware and software environment, the MPS had many successes. It was used to help ready the EOC for the first compatibility tests of the ground system with the spacecraft.

It was instrumental in identifying commanding problems to the spacecraft and through the ground network. The EOS integration and test team used the MPS extensively to test the EOC and conduct a series of ground system test cases, and the MPS was frequently credited for helping to identify problems and then verify their resolution. In succession, there were three control centers. The MPS was developed with the first EOC, and then was in place for the two follow-on systems. The MPS proved to be a robust test tool for development and test, and its availability contributed significantly to the rapid integration of the two later, mostly COTS-based control center alternatives.

On the strength of the successful simulator for AM-1, the development team was requested to build additional simulators for PM-1, due to launch in December 2000, and CHEM-1, due to launch in June 2002. Lessons learned from the MPS/AM-1 experience include the following:

- If the MPS had been cheaper, users would have had more units.
- A simpler architecture would have created fewer operational problems for the many casual users, especially when the system was used remotely.
- Users need more automation to assist in repeatability of tests.
- Users need more dynamic data.
- MPS developers and users need better test tools for troubleshooting.
- The system must capture and log all data, not just good data, needed for problem identification.
- Building in flexibility allows the simulator to be used for "what-if" exercises.

The SCE will apply this experience and insight to build the next simulator for the PM-1 project, the first one targeted for the new SIMSS architecture. The goal is to produce an operational system within a short timeframe at significantly reduced cost and, eventually, to produce the metrics to substantiate that claim.

EVOLUTION OF NEXT GENERATION SIMULATOR

The idea for the next generation of spacecraft simulators was formulated in a white paper produced by several staff members of the SCE in January 1998. The simulator development approach goals were to

- Reduce hardware costs
- Increase software development productivity
- Reduce development time
- Allow ease of COTS/GOTS integration where appropriate
- Develop a framework to support a component-based system and build a library of core capabilities that are extended where necessary to meet customer requirements
- Reduce test time with new module integration

A working group held a series of technical interchange meetings to evaluate several candidate platforms and operating systems. They recommended that the first target platform be a PC running Windows NT. The group derived a set of requirements for the simulator and began working on high-level designs for the user interface and simulator processor.

The new concept was proposed to the EOS project as a candidate architecture for the PM-1 simulator in November 1998, and the development team subsequently began working on a proof-of-concept prototype to determine whether the NT operating system could support basic simulator requirements. The prototype, which was demonstrated to the EOS project in February 1999, was intended as a throwaway; in actuality, it was enhanced and became an AM-1 portable simulator. It did successfully show that a low-cost Windows NT PC could be used for anticipated simulation and test capabilities.

The migration to a new platform, architecture, and development paradigm presented several challenges. This was a major initiative of the SCE and new personnel with the needed NT, C++, and Java skills had been hired to round out the development team. Experienced simulator developers without the object-oriented design and language skills were paired with ones who had the technology skills but not the simulator background. The development team received 2 weeks of training in object-oriented analysis and design techniques and C++ programming. Although some work on the baseline system had been done, work on the new simulator started in earnest in mid-March. A rapid application development approach was used, yet most in the group had never done this before. Two consultants from elsewhere within CSC were brought in to help adapt the methodology to the particular needs of the team. The results were astounding.

For two months there were twice weekly team meetings to discuss use cases, class diagrams, sequence diagrams, client- and server-side designs, message formats and types, graphical user interface (GUI) issues, coding standards, COTS product limitations, design and code reviews, and a number of other topics. On reflection, it seems disjointed; although, this was what the consultants told the team to expect. Out of this creative, sometimes chaotic, and spirited environment, and always under the watchful eye and guided direction of a talented system architect, the new concept and design for the next generation simulation system was turned from vision to reality. The first release was delivered in July.

The next generation simulation system has a dramatically different look and feel to the SCE simulators of the past – one ready to launch into the next millennium. As noted previously, this baseline system is called the Scalable, Integrated, Multimission Simulation Suite (SIMSS).

HARDWARE ARCHITECTURE

SIMSS runs on a standard Intel-based PC. The current target system is a Pentium III, 450 MHz processor with 512 K cache and 128 MB SDRAM. A commercial timing card provides an external clock source. An Ethernet card provides network communication capability and supports single or multiple LAN communication network interfaces. The system supports Industry Standard Architecture (ISA) and Peripheral Component Interconnection (PCI) buses. It is designed with extreme portability, remote connection capability, and cost-effectiveness in mind. Portable units are available for transport to ground stations and launch sites. Costs of both versions are comparable and are about $5,000. per unit for systems not requiring serial interface cards. Commercial serial data cards range in cost from $500 to $3,500. per card depending on the card functionality. Using the industry-standard PC platform makes a wider variety of COTS peripherals possible than ever before.

Windows NT 4.0 is the present operating system. It is a multi-threaded, preemptible operating system with performance sufficient to provide spacecraft simulation with near real time response. The network layer of the NT 4.0 operating system has proven to be reliable network software for the immediate application. It supports standard network protocols including TCP/IP, UDP/IP, and multicasting.

SIMSS SOFTWARE ARCHITECTURE

SIMSS is a module-based system. SIMSS provides an infrastructure to support all modules and provides the lower-level libraries and inter-module communication used by all modules. The primary module is the spacecraft simulator component (SIMSS/SC). All other modules are collectively referred to as network test tool modules (SIMSS/NeTT).

A client and server architecture is used. The GUI client is written in Java for cross platform compatibility. The client communicates with the server, which performs the actual simulation functions. The simulator and other server side components are written in C++ using a modular, object-oriented design. Anticipated reuse is predicated on the modularity and making the simulator data-driven as much as possible, using the contents of the project database to drive the simulation rather than custom code.

From the users' perspective, as seen in Figure 2, the SIMSS concept represents a set of modules that can be arranged in a graphical layout on the display. The user connects the modules to form a path that the data will follow, thus providing a flexible, viewable, and easily configurable system to meet a variety of simulation and testing needs. From the developers' perspective, as seen in Figure 3, the concept is one of developing generic functions in the core system and putting the mission-specific extensions outside of the core. Use of databases and user-provided configuration files minimize code changes for each new project.

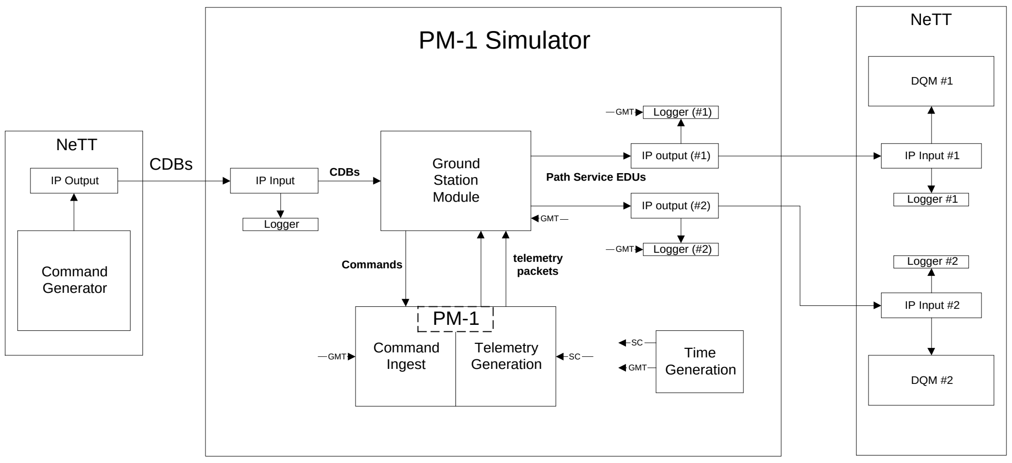

Figure 2. Sample Arrangement of SIMSS Modules

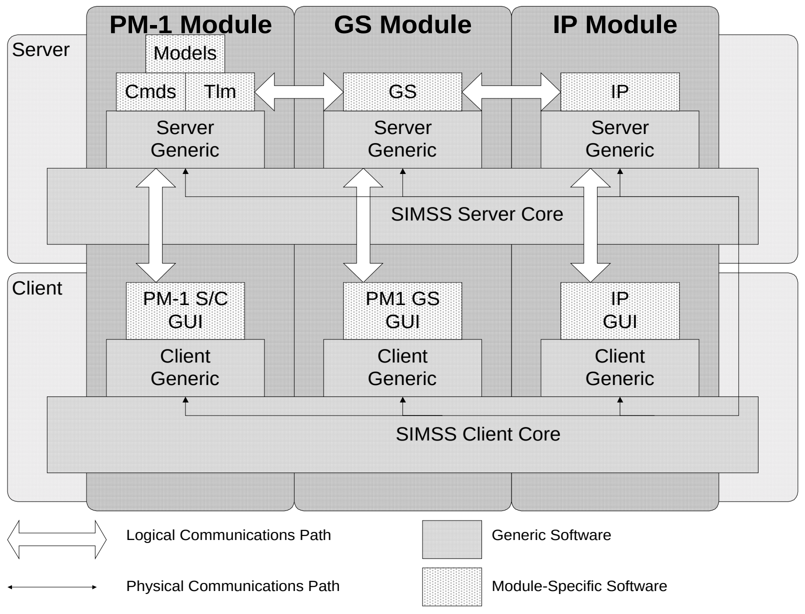

Figure 3. Internal Building Block Structure of Core and Mission-Specific SIMSS Components

Major functions are allocated to separate modules. The spacecraft (S/C) simulator module contains generic and mission-specific code. It generates telemetry, ingests and validates commands, uses the PDB if available, and supports mission-required simulation features, such as memory loads and dumps, stored command simulation, solid state recorder simulation, or other subsystem simulation.

The Ground Station (GS) module reformats data. In the EOS ground system, EDOS extracts commands from data blocks transmitted in User Datagram Protocol (UDP)/IP from the control center, and serializes the data for transfer and uplink to the spacecraft. For return link, EDOS receives serial clock and data, assembles data packets, quality checks the data, appends a header, and sends the data via UDP/IP to the control center. It also extracts the command link control word (CLCW) from telemetry and sends it as a separate packet to the control center in support of the CCSDS command verification protocol. Thus for SIMSS/PM-1, the GS module reformats the command and telemetry data simulating EDOS. For other missions, the GS module would provide comparable wrapper functions.

SIMSS interface modules handle data I/O. There are input and output modules for both IP and serial data interfaces. The user specifies IP addresses, port numbers, unicast/multicast option, and other configuration details. The serial modules, although not used for SIMSS/PM-1, contain setup parameters to control the serial interface card. Each data stream is connected separately to an interface module.

The other network test tool modules perform a variety of specialized functions. The log module can be inserted anywhere to write received data to a log file for later playback or analysis. Data logging is done on a per stream basis.

The data quality monitor (DQM) module is used to check the validity of selected header fields in received data. The module also counts number of received data units. This module is currently limited in functionality but will be enhanced to detect sequence counter discontinuities, flag erroneous data, and filter data on identification fields for logging or transfer, for example.

The command generator module is a generalized test function that creates and transmits spacecraft commands to the simulator for internal testing. This is a useful feature for simulator developers to verify the command handling of the simulator prior to its use with the control center.

BENEFITS OF SIMSS CONCEPT

Just the hardware cost savings going from the VME to PC is a significant benefit. Based on the reduced cost of the hardware platform alone, it will be possible for users to have multiple units available on site and to locate them at both development and operations facilities. Several copies of the simulator could be provided to control center software developers much earlier in the development life cycle where detection and correction of software errors is most cost effective. Meanwhile, FOT members could do their own testing in the operations environment, also possibly using multiple simulators. For EOS AM-1, there could have been 12 SIMSS/AM-type PC units for the hardware cost alone of one MPS VME-based unit.

The current PC costing trend should continue resulting in increased performance at lower hardware costs. There are also many more powerful and affordable peripheral devices such as writable CD-ROMs and zip drives available with PCs. Another benefit of the standard PC platform over the VME counterpart is that the PC is multipurpose. It can be used for other support tasks such as word processing or web browsing needed in everyday operations.

Good development tools are available for the Windows NT PC environment. Visual C/C++ is the key tool in use for server-side software. It supports object-oriented development and rich main stream API calls. It also supports Dynamic Link Library capabilities, which allow for object loading and unloading while running without modification to the main server program. Objects can be independent of the baseline software, so that mission-specific objects can be designed and modified without requiring changes to the core, baseline system. Visual Café is used for developing the Java GUI client. Design and configuration management tools also have resulted in additional productivity gains.

Once the baseline SIMSS is fully in place, significant reductions in development time will be possible. This modular system can be added to and changed easily by recompiling and reconfiguring only those modules that need to be customized. In addition, users will be able to tailor configurations to suit their needs and more easily interface with other equipment. Time saving benefits have already been realized for other follow-on projects that were able to build upon the existing library of reusable components. And this will only grow over time. For future missions, a project would define its mission-specific requirements and configuration. These requirements can then be used as a basis for adapting the generic modular system to support the mission requirements, in effect “plugging in” the specific requirements into the system. For the EOS project, the same spacecraft manufacturer is building both the PM-1 and CHEM-1 satellites. Extremely high code reuse and cost savings are anticipated for the follow-on CHEM simulator development effort.

Once established, the goal for this new baseline will be to allow a small team to produce a low-fidelity spacecraft simulator in a matter or weeks to months, a medium-fidelity spacecraft simulator in less than a year, or a high-fidelity spacecraft simulator that models the spacecraft as closely as is needed. The baseline system also provides for an assortment of integrated communications test tools that can be used to check out various ground system elements or to translate or convert spacecraft communications data from one format to another.

FUTURE ENHANCEMENTS

Much of the core SIMSS infrastructure has been developed. Initial PM-1 telemetry and commanding data formats are supported. Several more SIMSS/PM-1 releases have been planned with project personnel and simulator users based on ground system development schedules and priorities. Examples of capabilities targeted for future SIMSS/PM-1 releases include: playback and interleaving of user-provided data files; scenarios files that provide a powerful scripting feature to support variable data generation and automated test execution; model generators to simulate more realistic data; and expanded error insertion.

A goal is to integrate SIMSS with commercially available and government developed control centers. Just as the control centers contain generic features to support a wide range of missions with minimal customization, so too is the intent for SIMSS. SIMSS could be a valuable tool for ground system integration and testing for a variety of spacecraft supported by the COTS/GOTS control centers.

In addition, numerous other enhancements are planned for the core SIMSS system. SIMSS will be ported to the Linux operating system to provide a PC system for operation in Unix-based environments. Additional efforts will be taken to use COTS libraries to eliminate platform dependencies. Also under investigation are distributed object enhancements including CORBA and DCOM, and ways to incorporate the Defense Modeling and Simulation Office’s High Level Architecture (HLA).

CONCLUSION

Significant strides have been made towards reaching each of the simulator development goals established at the onset on this project. There have been substantial reductions in hardware costs. Software developer productivity, experience with state-of-the-practice skills, and morale have all improved. COTS products and new technology have been integrated where appropriate; others are being evaluated for later infusion. The framework is nearly constructed and the library of reusable modules will continue to grow. There is every indication that development time will decrease once the baseline system is complete. Users will be able to opt for lower costs or enhanced capabilities when compared with the SCE systems of the past.

Reference

The Simulations and Test Working Group, *White Paper on Developing the Next Generation of Simulation and Testing Tools*, January, 1998, http://cmex.gsfc.nasa.gov/, Simulations Center of Excellence at NASA Goddard Space Flight Center.